#########################################################################

# Summary of VMD scripts used in

#

#Cheng,MH, Torres-Salazar, D, Gonzalez-Suarez, AD, Amara, SG & Bahar, I.

#“Substrate transport and anion permeation proceed through distinct #pathways in glutamate transporters”. eLife, in press (2017).

#########################################################################

1. Calculate the residue solvent accessibility surface area (SASA)

# EQUIREMENTS: VMD Version 1.8.1 or greater

# DESCRIPTION:

# load all MD snapshots to VMD

# EXAMPLE USAGE:

# source calc\_SASA.tcl

calc\_SASA.tcl

=========================================================================set nf [molinfo top get numframes]

set outfile [open SASA/V51\_3V8G\_E02\_A\_B\_C w];

for {set i 0 } { $i < $nf } { incr i } {

set selA [atomselect top "chain A" frame $i ]

set selA\_51 [atomselect top "chain A and resid 51" frame $i ]

set SASA\_A [measure sasa 1.4 $selA -restrict $selA\_51]

set selB [atomselect top "chain B" frame $i ]

set selB\_51 [atomselect top "chain B and resid 51" frame $i ]

set SASA\_B [measure sasa 1.4 $selB -restrict $selB\_51]

set selC [atomselect top "chain C" frame $i ]

set selC\_51 [atomselect top "chain C and resid 51" frame $i ]

set SASA\_C [measure sasa 1.4 $selC -restrict $selC\_51]

puts $outfile "$SASA\_A $SASA\_B $SASA\_C"

}

close $outfile

=========================================================================

#########################################################################

2. Calculate the distance between HP2 and S65 in Gltph.

# EQUIREMENTS: VMD Version 1.8.1 or greater

# DESCRIPTION:

# calculate the distance between HP1 and HP2 loops

# EXAMPLE USAGE:

# source big\_dcd\_distz\_HP2\_S65.tcl

#########################################################################

big\_dcd\_distz\_HP2\_S65.tcl

========================================================================= proc mydist { frame } {

global ref sel all file\_A file\_B file\_C sel\_1\_A sel\_2\_A sel\_1\_B sel\_2\_B sel\_1\_C sel\_2\_C

# $all move [measure fit $sel $ref]

set cent\_sel1\_A [measure center $sel\_1\_A]

set cent\_sel2\_A [measure center $sel\_2\_A]

set sel1\_z\_A [lindex $cent\_sel1\_A 2]

set sel2\_z\_A [lindex $cent\_sel2\_A 2]

set dist\_z\_A [expr $sel2\_z\_A-$sel1\_z\_A]

puts $file\_A "$dist\_z\_A"

set cent\_sel1\_B [measure center $sel\_1\_B]

set cent\_sel2\_B [measure center $sel\_2\_B]

set sel1\_z\_B [lindex $cent\_sel1\_B 2]

set sel2\_z\_B [lindex $cent\_sel2\_B 2]

set dist\_z\_B [expr $sel2\_z\_B-$sel1\_z\_B]

puts $file\_B "$dist\_z\_B"

set cent\_sel1\_C [measure center $sel\_1\_C]

set cent\_sel2\_C [measure center $sel\_2\_C]

set sel1\_z\_C [lindex $cent\_sel1\_C 2]

set sel2\_z\_C [lindex $cent\_sel2\_C 2]

set dist\_z\_C [expr $sel2\_z\_C-$sel1\_z\_C]

puts $file\_C "$dist\_z\_C"

}

proc bigdcd { script type args } {

global bigdcd\_frame bigdcd\_proc bigdcd\_firstframe vmd\_frame bigdcd\_running

set bigdcd\_running 1

set bigdcd\_frame 0

set bigdcd\_firstframe [molinfo top get numframes]

set bigdcd\_proc $script

# backwards "compatibility". type flag is omitted.

if {[file exists $type]} {

set args [linsert $args 0 $type]

set type auto

}

uplevel #0 trace variable vmd\_frame w bigdcd\_callback

foreach dcd $args {

if { $type == "auto" } {

mol addfile $dcd waitfor 0

} else {

mol addfile $dcd type $type waitfor 0

}

}

after idle bigdcd\_wait

}

proc bigdcd\_callback { tracedvar mol op } {

global bigdcd\_frame bigdcd\_proc bigdcd\_firstframe vmd\_frame

set msg {}

# If we're out of frames, we're also done

# AK: (can this happen at all these days???). XXX

set thisframe $vmd\_frame($mol)

if { $thisframe < $bigdcd\_firstframe } {

puts "end of frames"

bigdcd\_done

return

}

incr bigdcd\_frame

if { [catch {uplevel #0 $bigdcd\_proc $bigdcd\_frame} msg] } {

puts stderr "bigdcd aborting at frame $bigdcd\_frame\n$msg"

bigdcd\_done

return

}

animate delete beg $thisframe end $thisframe $mol

return $msg

}

proc bigdcd\_done { } {

global bigdcd\_running

if {$bigdcd\_running > 0} then {

uplevel #0 trace vdelete vmd\_frame w bigdcd\_callback

puts "bigdcd\_done"

set bigdcd\_running 0

}

}

proc bigdcd\_wait { } {

global bigdcd\_running bigdcd\_frame

while {$bigdcd\_running > 0} {

global bigdcd\_oldframe

set bigdcd\_oldframe $bigdcd\_frame

# run global processing hooks (including loading of scheduled frames)

display update ui

# if we have read a new frame during then the two should be different.

if { $bigdcd\_oldframe == $bigdcd\_frame } {bigdcd\_done}

}

}

#input psf name of simulation system

set inputpsf 3V8G\_BlooponA\_CloopIFS\_pkaCharge\_NACL01M\_POPCC36.psf

#input pdb name of simulation system

set refpdb 3V8G\_BlooponA\_CloopIFS\_pkaCharge\_NACL01M\_POPCC36.pdb

#output file name for chain B

set file\_B [open ./DIST\_Z/distz\_EN03\_HP2\_S65\_B\_pKA a]

#output file name for chain A

set file\_A [open ./DIST\_Z/distz\_EN03\_HP2\_S65\_A\_pKA a]

#output file name for chain C

set file\_C [open ./DIST\_Z/distz\_EN03\_HP2\_S65\_C\_pKA a]

set mol [mol new $inputpsf type psf waitfor all]

set all [atomselect $mol all]

set ref [atomselect $mol "name CA" frame 0]

set sel [atomselect $mol "name CA"]

set sel\_1\_A [atomselect $mol "chain A and resid 335 to 370"]

set sel\_2\_A [atomselect $mol "chain A and resid 65"]

set sel\_1\_B [atomselect $mol "chain B and resid 335 to 370"]

set sel\_2\_B [atomselect $mol "chain B and resid 65"]

set sel\_1\_C [atomselect $mol "chain C and resid 335 to 370"]

set sel\_2\_C [atomselect $mol "chain C and resid 65"]

mol addfile $refpdb waitfor all

#input dcd file name

bigdcd mydist dcd MD\_Cl\_2\_EN03\_NPT10/3V8G\_BlooponA\_CloopIFS\_pkaCharge\_MD\_TMD600\_From6ns\_EN03\_NPT10.dcd

bigdcd\_wait

close $file\_A

close $file\_B

close $file\_C

=========================================================================

#########################################################################

3. Calculate the distance between HP1 and S65 in Gltph.

# EQUIREMENTS: VMD Version 1.8.1 or greater

# DESCRIPTION:

# calculate the distance between HP1 and HP2 loops

# EXAMPLE USAGE:

# source big\_dcd\_distz\_HP1\_S65.tcl

#########################################################################

big\_dcd\_distz\_HP1\_S65.tcl

=========================================================================

proc mydist { frame } {

global ref sel all file\_A file\_B file\_C sel\_1\_A sel\_2\_A sel\_1\_B sel\_2\_B sel\_1\_C sel\_2\_C

set cent\_sel1\_A [measure center $sel\_1\_A]

set cent\_sel2\_A [measure center $sel\_2\_A]

set sel1\_z\_A [lindex $cent\_sel1\_A 2]

set sel2\_z\_A [lindex $cent\_sel2\_A 2]

set dist\_z\_A [expr $sel2\_z\_A-$sel1\_z\_A]

puts $file\_A "$dist\_z\_A"

set cent\_sel1\_B [measure center $sel\_1\_B]

set cent\_sel2\_B [measure center $sel\_2\_B]

set sel1\_z\_B [lindex $cent\_sel1\_B 2]

set sel2\_z\_B [lindex $cent\_sel2\_B 2]

set dist\_z\_B [expr $sel2\_z\_B-$sel1\_z\_B]

puts $file\_B "$dist\_z\_B"

set cent\_sel1\_C [measure center $sel\_1\_C]

set cent\_sel2\_C [measure center $sel\_2\_C]

set sel1\_z\_C [lindex $cent\_sel1\_C 2]

set sel2\_z\_C [lindex $cent\_sel2\_C 2]

set dist\_z\_C [expr $sel2\_z\_C-$sel1\_z\_C]

puts $file\_C "$dist\_z\_C"

}

proc bigdcd { script type args } {

global bigdcd\_frame bigdcd\_proc bigdcd\_firstframe vmd\_frame bigdcd\_running

set bigdcd\_running 1

set bigdcd\_frame 0

set bigdcd\_firstframe [molinfo top get numframes]

set bigdcd\_proc $script

# backwards "compatibility". type flag is omitted.

if {[file exists $type]} {

set args [linsert $args 0 $type]

set type auto

}

uplevel #0 trace variable vmd\_frame w bigdcd\_callback

foreach dcd $args {

if { $type == "auto" } {

mol addfile $dcd waitfor 0

} else {

mol addfile $dcd type $type waitfor 0

}

}

after idle bigdcd\_wait

}

proc bigdcd\_callback { tracedvar mol op } {

global bigdcd\_frame bigdcd\_proc bigdcd\_firstframe vmd\_frame

set msg {}

# If we're out of frames, we're also done

# AK: (can this happen at all these days???). XXX

set thisframe $vmd\_frame($mol)

if { $thisframe < $bigdcd\_firstframe } {

puts "end of frames"

bigdcd\_done

return

}

incr bigdcd\_frame

if { [catch {uplevel #0 $bigdcd\_proc $bigdcd\_frame} msg] } {

puts stderr "bigdcd aborting at frame $bigdcd\_frame\n$msg"

bigdcd\_done

return

}

animate delete beg $thisframe end $thisframe $mol

return $msg

}

proc bigdcd\_done { } {

global bigdcd\_running

if {$bigdcd\_running > 0} then {

uplevel #0 trace vdelete vmd\_frame w bigdcd\_callback

puts "bigdcd\_done"

set bigdcd\_running 0

}

}

proc bigdcd\_wait { } {

global bigdcd\_running bigdcd\_frame

while {$bigdcd\_running > 0} {

global bigdcd\_oldframe

set bigdcd\_oldframe $bigdcd\_frame

# run global processing hooks (including loading of scheduled frames)

display update ui

# if we have read a new frame during then the two should be different.

if { $bigdcd\_oldframe == $bigdcd\_frame } {bigdcd\_done}

}

}

set inputpsf 3V8G\_BlooponA\_CloopIFS\_pkaCharge\_NACL01M\_POPCC36.psf

set refpdb 3V8G\_BlooponA\_CloopIFS\_pkaCharge\_NACL01M\_POPCC36.pdb

set file\_B [open ./DIST\_Z/distz\_EN03\_HP1\_S65\_B\_pKA a]

set file\_A [open ./DIST\_Z/distz\_EN03\_HP1\_S65\_A\_pKA a]

set file\_C [open ./DIST\_Z/distz\_EN03\_HP1\_S65\_C\_pKA a]

set mol [mol new $inputpsf type psf waitfor all]

set all [atomselect $mol all]

set ref [atomselect $mol "name CA" frame 0]

set sel [atomselect $mol "name CA"]

set sel\_1\_A [atomselect $mol "chain A and resid 258 to 292"]

set sel\_2\_A [atomselect $mol "chain A and resid 65"]

set sel\_1\_B [atomselect $mol "chain B and resid 258 to 292"]

set sel\_2\_B [atomselect $mol "chain B and resid 65"]

set sel\_1\_C [atomselect $mol "chain C and resid 258 to 292"]

set sel\_2\_C [atomselect $mol "chain C and resid 65"]

mol addfile $refpdb waitfor all

bigdcd mydist dcd MD\_Cl\_2\_EN03\_NPT10/3V8G\_BlooponA\_CloopIFS\_pkaCharge\_MD\_TMD600\_From6ns\_EN03\_NPT10.dcd

bigdcd\_wait

close $file\_A

close $file\_B

close $file\_C

=========================================================================

#########################################################################

#4. calculate and visulize the channel pore

# hole: a script for running the HOLE program written by Oliver Smart

# URL: http://www.bip.bham.ac.uk/osmart/hole/top.html

#

# Here's example usage:

#

# source hole2.tcl

# mol load pdb temp.pdb

# set sel [atomselect top all]

# Hole::runhole $sel

#

# Justin Gullingsrud

# jgulling@mccammon.ucsd.edu

# 25 October 2003

#########################################################################

hole2.tcl

=========================================================================

namespace eval Hole {

# Customize the following lines to set the paths to the hole executable and

# the radius file.

variable holebin /usr/local/hole2/exe/

variable holerad /usr/local/hole2/rad/simple.rad

# Customize the following to set default values. See the runhole

# comments for what these parameters do.

variable cvect [list 0. 0. 1.]

variable cpoint [list -13.5 37.02. -11.84]

variable sample 1.0.

variable endrad 15.

}

# Routine for calling hole. Pass molid, frame, and optional keyword

# arguments:

# -cvect {x y z} vector parallel to channel axis

# -cpoint {x y z} a point in the pore

# -sample s distance between samples

# Returns a list whose elements are of the form {z r resname resid},

# where z is the channel coordinate, r is the radius at that coordinate,

# and resname and resid give the identity of the residue nearest the center

# of the pore at that coordinate.

proc Hole::runhole {holeoutput sel args} {

variable holebin

variable holerad

variable cvect

variable cpoint

variable sample

variable endrad

# parse options

foreach { opt val } $args {

switch $opt {

-cvect {

if { [llength $cvect] != 3 } {

error "-cvect must have three elements"

}

set cvect $val

}

-cpoint {

if { [llength $cpoint] != 3 } {

error "-cpoint must have three elements"

}

set cpoint $val

}

-sample { set sample $val }

-endrad { set endrad $val }

default {

error "Unknown option '$opt'"

}

}

}

if { [$sel num] < 3 } {

error "Not enough atoms ([$sel num]) found in selection."

}

# write coordinates to files

set pdb tmpholeinputfiles.pdb

$sel writepdb $pdb

# construct HOLE input string

set str "\ncoord $pdb\n"

append str "radius $holerad\n"

append str "cvect $cvect\n"

append str "cpoint $cpoint\n"

append str "sample $sample\n"

append str "endrad $endrad\n"

append str "SPHPDB $holeoutput.sph\n"

# Call HOLE and collect output

puts "Calling HOLE..."

flush stdout

set result [exec "$holebin\hole" "<< $str"]

set rawdata\_z [list]

set rawdata\_r [list]

set rawdata\_resname [list]

set rawdata\_resid [list]

set fileid [open $holeoutput.xls a+ 0777]

set lines [split $result \n]

set n [llength $lines]

for { set i 0 } { $i < $n } { incr i } {

set line [lindex $lines $i]

if { [string first highest $line] != -1 } {

incr i

foreach { at point x y z } [lindex $lines $i] { break }

incr i

set line [lindex $lines $i]

set r [string trim [string range $line 22 30]]

set aname [string trim [string range $line 31 37]]

set resname [string trim [string range $line 38 40]]

set resid [string trim [string range $line 44 end]]

puts $fileid "$z $r $resname $resid"

}

}

close $fileid

file delete tmpholeinputfiles.pdb

puts "Calling HOLE Sph\_Process"

catch {exec $holebin\sph\_process -dotden 15 -color 2 3 -sos $holeoutput.sph $holeoutput.sos} fid

set fileid2 [open $holeoutput.sph\_process\_log.txt a+ 0777]

puts $fileid2 $fid

close $fileid2

puts "Calling HOLE Sos\_Triangle"

catch {exec $holebin\sos\_triangle -s -v < $holeoutput.sos > $holeoutput.vmd} fid2

set fileid3 [open $holeoutput.sos\_triangle\_log.txt a+ 0777]

puts $fileid3 $fid2

close $fileid3

puts "Finished"

}

========================================================================